National Institute of Standards and Technology
U.S. Department of Commerce

Special Publication 800-81-2
Unofficial Errata
Version 1.0
4/21/2014

This is an unofficial update/changes to NIST Special Publication 800-81-2. The Special Publication was released while the Internet community is still in the process on developing best common practices and improving the security of DNSSEC. These changes to not alter the main checklist items in the guide only refine the text to keep the document up to date with respects to ongoing work from the Internet community.

This document will be updated on an irregular basis as new material is produced, best common practices are refined, and operators gain more experience with DNSSEC. This document will remain "unofficial" in that it is not part of the official Special Publication errata process for now. An official version may be published in the future.

In the text below, the text that appears in the NIST SP 80-81-2 guide is marked in blue, changes in black, and any text that appears in *italics* describes the reasoning behind the change or addition if necessary.

2.1 What is the Domain Name System (DNS)?

The Internet is the world's largest computing network, with more than 580 million users. From the perspective of a user, each node or resource on this network is identified by a unique name: the domain name. Some examples of Internet resources are:

- Web servers—for accessing Web sites
- Mail servers—for delivering e-mail messages
- Application servers—for accessing software systems and databases remotely.

From the perspective of network equipment (e.g., routers) that routes communication packets across the Internet, however, the unique resource identifier is the Internet Protocol (IPv4 or IPv6) address, represented as a series of four numbers separated by dots (e.g., 123.67.43.254 is an IPv4 address). To access Internet resources by user-friendly domain names rather than these IP addresses, users need a system that translates these domain names to IP addresses and back. This translation is the primary task of an engine called the *Domain Name System* (DNS).

Users access an Internet resource (e.g., a Web server) through the corresponding client or user program (e.g., a Web browser) by typing the domain name. To contact the Web server and retrieve the appropriate Web page, the browser needs the corresponding IP address. It calls DNS to provide this information. This function of mapping domain names to IP addresses is called *name resolution*. The protocol that DNS uses to perform the name resolution function is called the DNS protocol.

The DNS function described above includes the following building blocks. First, DNS should have a data repository to store the domain names and their associated IP addresses. Because the number of domain names is large, scalability and performance considerations dictate that it should be distributed. The domain names may even need to be replicated to provide fault tolerance. Second, there should be software that manages this repository and provides the name resolution function. These two functions (managing the domain names repository and providing name resolution service) are provided by the primary DNS component, the *name server*. There are many categories of name servers, distinguished by type of data served and functions performed. To access the services provided by a DNS name server on behalf of user programs, there is another component of DNS called the *resolver*. There are two primary categories of resolvers (caching/recursive/resolving name server and stub resolver),[1] distinguished by functionality. The communication protocol; the various DNS components; the policies governing the configuration of these components; and procedures for creation, storage, and usage of domain names constitute the DNS infrastructure

7.2.5 Dedicated Name Server Instance for Each Function

Only the example configuration code

```
options {
      recursion no;
      minimal-response yes;
      additional-from-auth no;
      additional-from-cache no;
};
```

The other options help authoritative server operations. `minimal-response` configures the server to only put RRsets in the Authoritative and Additional section of a DNS reply when needed, otherwise no RRsets are included. This saves space in a response as well as reduces the work done by the server. `additional-from-auth` and `additional-from-cache` restricts the server as to where it gets RRsets to put in the Authoritative and Additional sections when required. These options configure the server to not include data that may be in other authoritative zones served by the server or any cached data in a DNS response. Data in a DNS response will only include authoritative data from the queried zone.

8.1.1.1 Restricting Recursive Queries (a special case under DNS

Query/Response)

Only the example configuration code

The server-wide option would be to restrict all queries to the list of clients and only those queries form clients that request recursion:

```
     options {
          allow-query { internal_hosts; };
          - or –
          allow-recursion { internal_hosts; };
          match-recursive-only yes;
    };
```

8.2.4 Instructing Name Servers to Use Keys in All Transactions

The command to instruct the server to use the key in all transactions (DNS query/response, zone transfer, dynamic update, etc.) is as follows:

```
     server 192.249.249.1 {
           keys { ns1-ns2.example.com.; };
     };
```

The same statement can be used as an entry in an `acl` statement as well:

```
     acl key_acl {
          key ns1-ns2.example.com.;
     };
```

The example is missing the key statement word.

8.2.6 Securing Zone Transfers using TSIG

Only the example configuration code

```
     zone "example.com" {
         type master;
         file "zonedb.example.com";
         allow-transfer { key ns1-ns2.example.com.; };
     };
```

In the original example, there were curly brackets around the key name in the allow-transfer statement. They should not have been there.

8.2.9 Configuring Fine-Grained Dynamic Update Restrictions Using TSIG/SIG(0) Keys

The allow-update substatement specifies dynamic update restrictions based on the originators of dynamic update requests (a specific set of hosts identified by IP address or holding a TSIG/SIG(0) key) but not the contents of the zone records. To specify dynamic update access (grant or deny) restrictions based on a combination of domain/subdomain names and RR types (A, MX, NS, etc.), BIND 9 and later versions provide the update-policy substatement within the zone statement. The update-policy substatement bases these restrictions on the TSIG/SIG(0) key. In other words, the update-policy statement specifies which TSIG/SIG(0) keys (or holders of keys) are allowed to perform dynamic updates on which domains/subdomains and RR types within that domain/subdomain. In the text below, either TSIG or SIG(0) keys could be used, but TSIG is the most common method as it has the easiest setup and will be used in the text.

9.3 Generation of Public Key-Private Key Pair (DNSSEC-OP1)

Final paragraph:

The use of RSA in DNSSEC is approved until the year 2015 within the US Federal Government. Its successor, Elliptic Curve Cryptography (ECC), is now specified in the DNSSEC [RFC6605] and already present in major cryptographic libraries. USG DNS administrators should plan to migrate to the use of ECDSA (or similar) when it becomes available in DNSSEC components. ECC has an advantage of having the same perceived strength as RSA with a smaller key size. This means that the ZSK can be the same size as the KSK and have a longer cryptoperiod than a 1024 bit RSA ZSK.

10.1 Choosing Parameter Values in SOA RR

- **Expire Value.** The *expire value* is the length of time a secondary server should consider the zone information valid if it can no longer reach the primary server to refresh. This field allows secondary servers to continue to operate until network disruptions are resolved. This value depends on the frequency of changes to the zone and the reliability of the connection between name servers. When the zone is signed, the expire value should be the same as the lowest signature expiration value in the zone. This is to avoid having name servers serving a copy of the zone with expired signatures. It should be a multiple of the refresh value and possibly set to as long as 2 to 4 weeks, or the value of the (lowest) signature expiration value.